(12) UK Patent Application (19) GB (11) 2 271 251 (13) A

(43) Date of A Publication 06.04.1994

(21) Application No 9220729.9

(22) Date of Filing 01.10.1992

(71) Applicant(s)
Digital Equipment International Limited

(Incorporated in Switzerland)

1 Grand Places, 1700 Fribourg, Switzerland

(72) Inventor(s)
Neil Alasdair James Jarvis

(74) Agent and/or Address for Service
Eric Potter Clarkson
St Mary's Court, St Mary's Gate, NOTTINGHAM, NG1 1LE, United Kingdom

(51) INT CL[5]
H04L 12/28 , G06F 13/42 , H04L 7/00

(52) UK CL (Edition M)
H4P PPF

(56) Documents Cited
EP 0135764 A2

(58) Field of Search
UK CL (Edition L) G4A AFT , H4K KTA KTX , H4P PDRX PPF PSEX PSX
INT CL[5] G06F 13/42 , H04L 7/00 7/10 12/28 12/42

(54) Timer synchronisation in data communications networks

(57) A method for the synchronisation of two processors in a network in which transmission times of communications between network nodes are variable, but timers may only be adjusted in one direction, e.g. incremented. The method comprises the steps of:

50) a first timer (M) issuing to a second timer (S) the current value of said first timer (M_0);

52) said second timer (S) comparing said first timer value (M_0) with the current value of said second timer (S_0);

60) if said second timer value (S_0) is less than said first timer value (M_0), said second timer (S) incrementing said current second timer value (S_0) to a new second timer value (S_1) equal said first timer value (0); and

62) if said second timer value (S_0) is greater than said first timer value (M_0), said second timer (S) issuing to said first timer (M) the current value of said second timer (S_0).

An alternative embodiment allows for the case where timers may only be decreased in value.

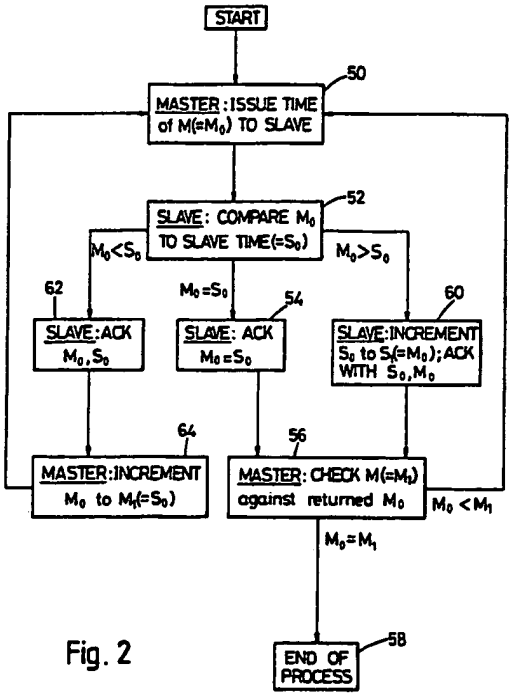

Fig. 2

GB 2 271 251 A

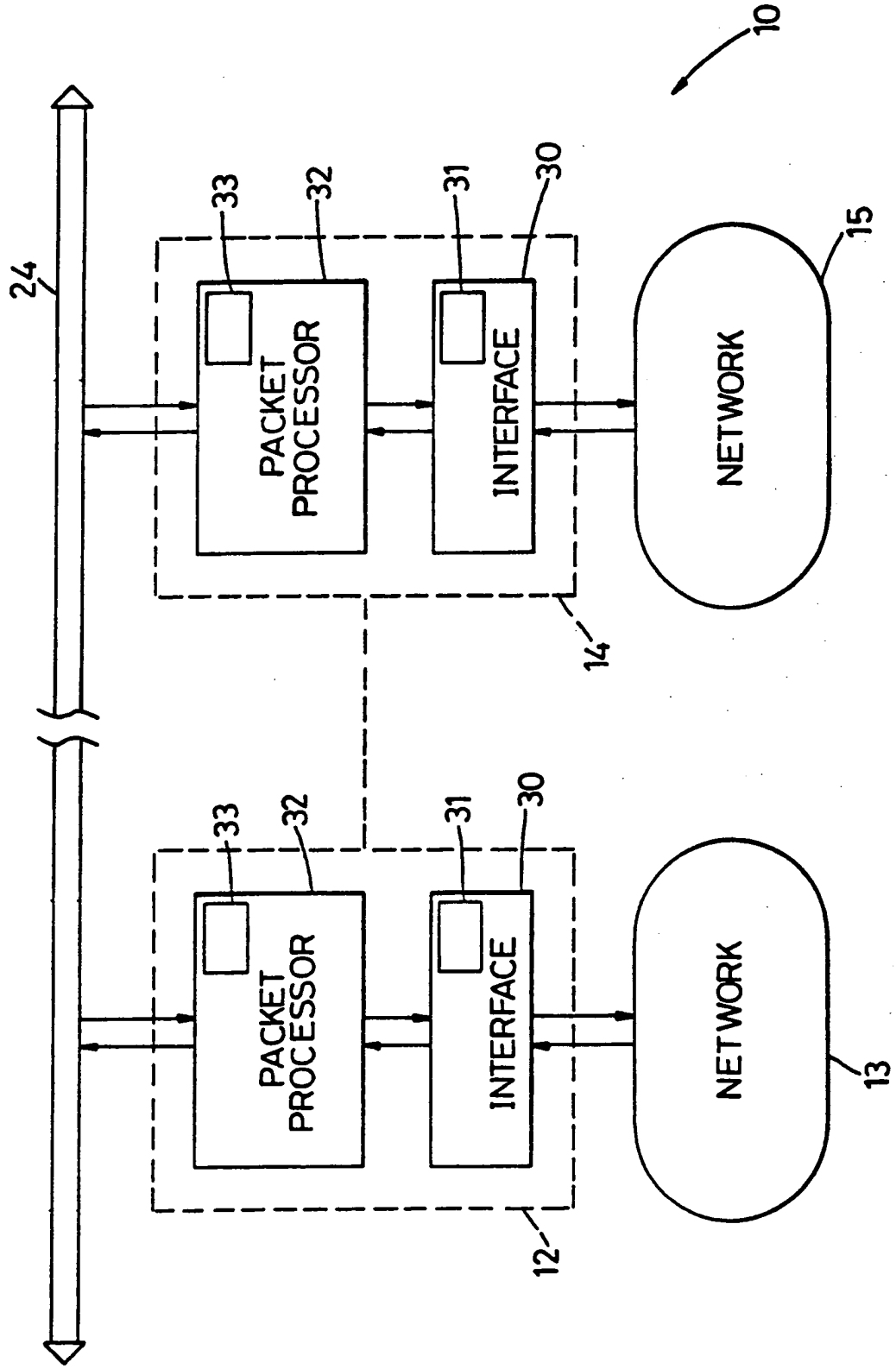

Fig. 1

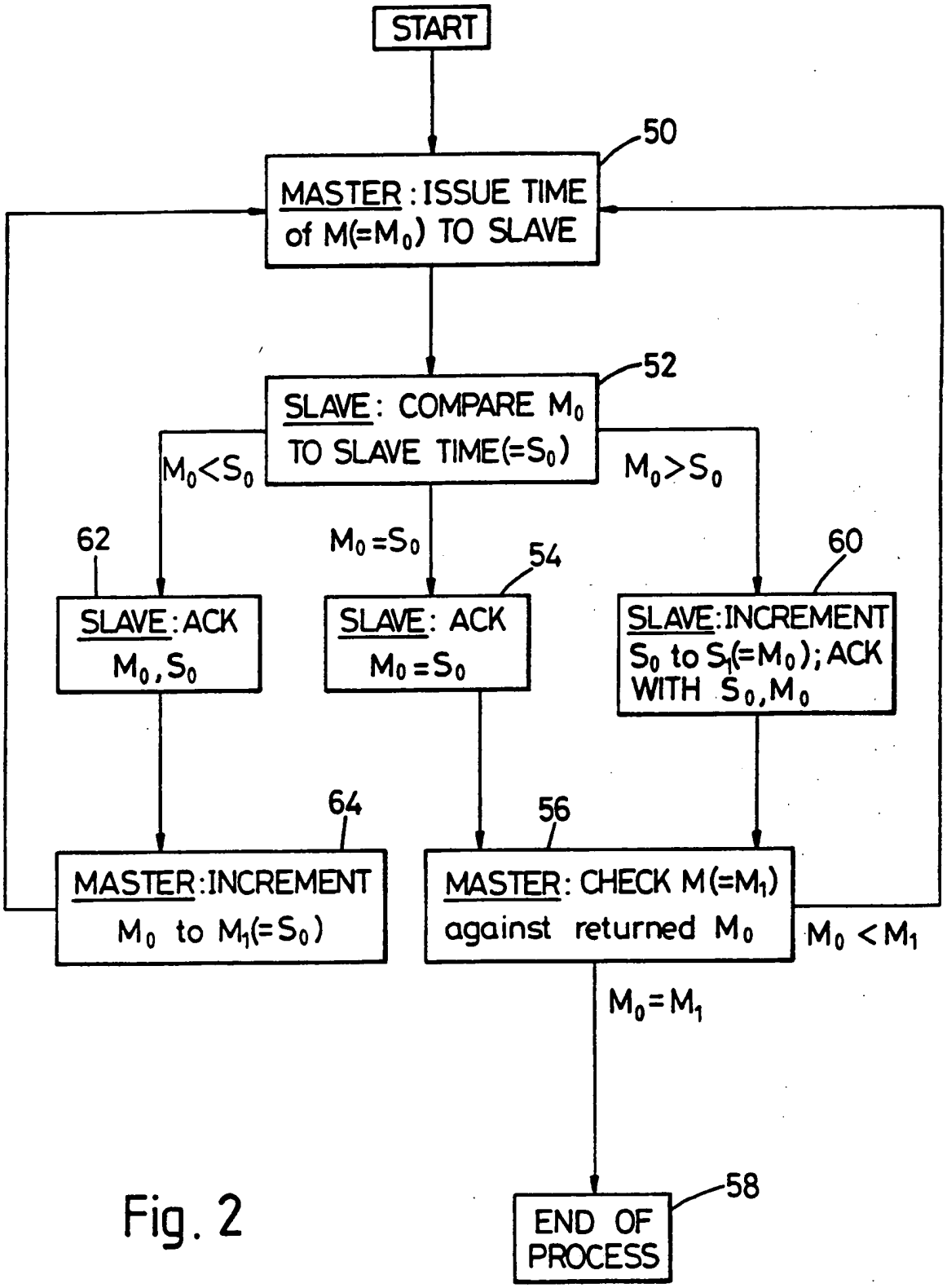
START
50
MASTER: ISSUE TIME of M(=M0) TO SLAVE
52
SLAVE: COMPARE M0 TO SLAVE TIME(=S0)
M0<S0
M0=S0
M0>S0
62
54
60
SLAVE: ACK M0, S0
SLAVE: ACK M0=S0
SLAVE: INCREMENT S0 to S1(=M0); ACK WITH S0, M0
64
56
MASTER: INCREMENT M0 to M1(=S0)
MASTER: CHECK M(=M1) against returned M0
M0<M1
M0=M1
58
END OF PROCESS

Fig. 2

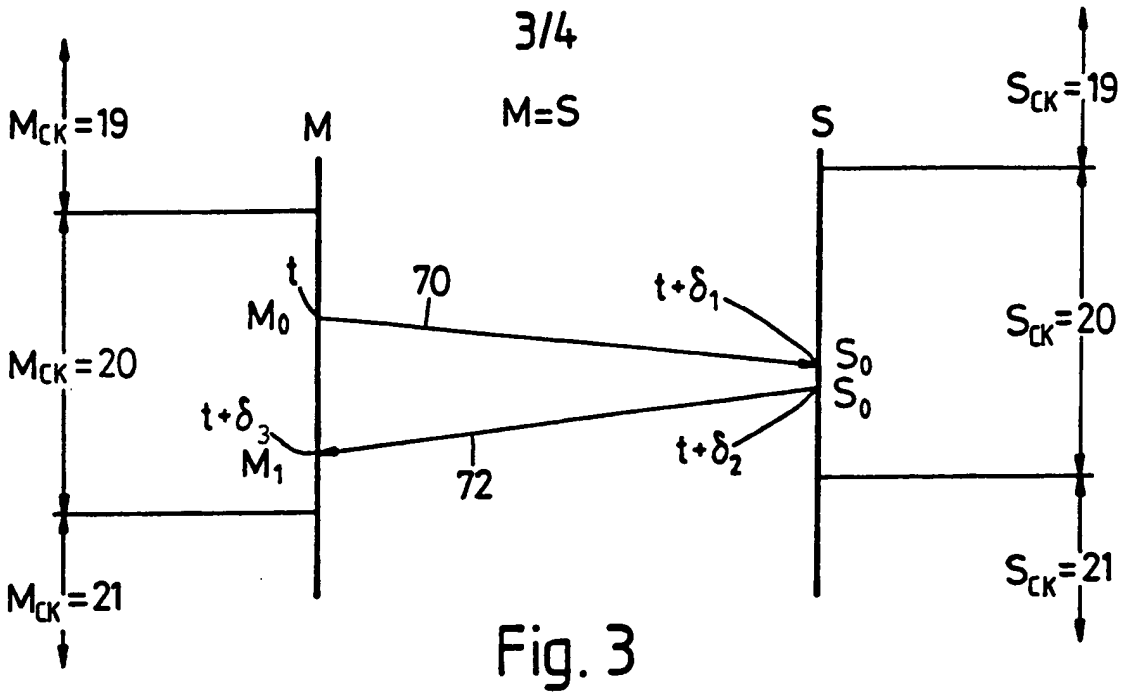

Fig. 3

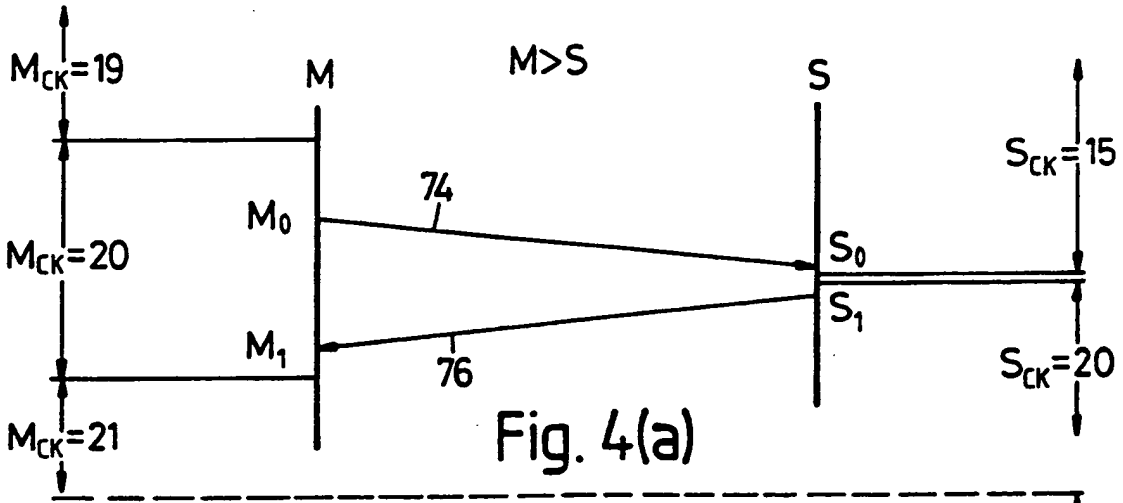

Fig. 4(a)

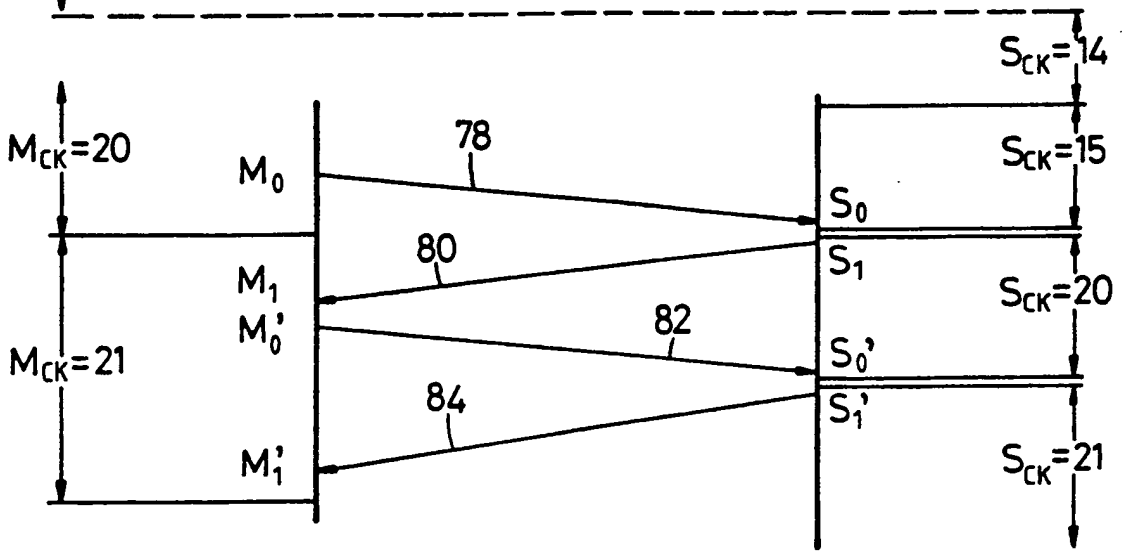

Fig. 4(b)

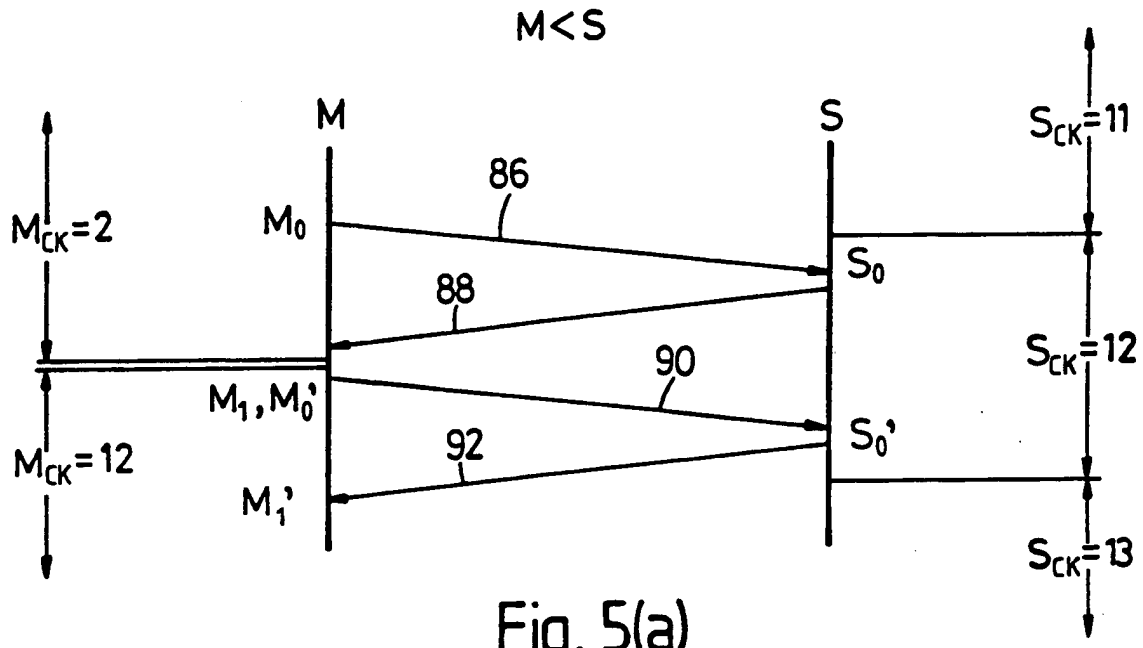
M<S
M
S
$S_{CK}=11$
86
$M_{CK}=2$
M_0
S_0
88
$S_{CK}=12$
90
M_1, M_0'
S_0'
92
$M_{CK}=12$
M_1'
$S_{CK}=13$

Fig. 5(a)

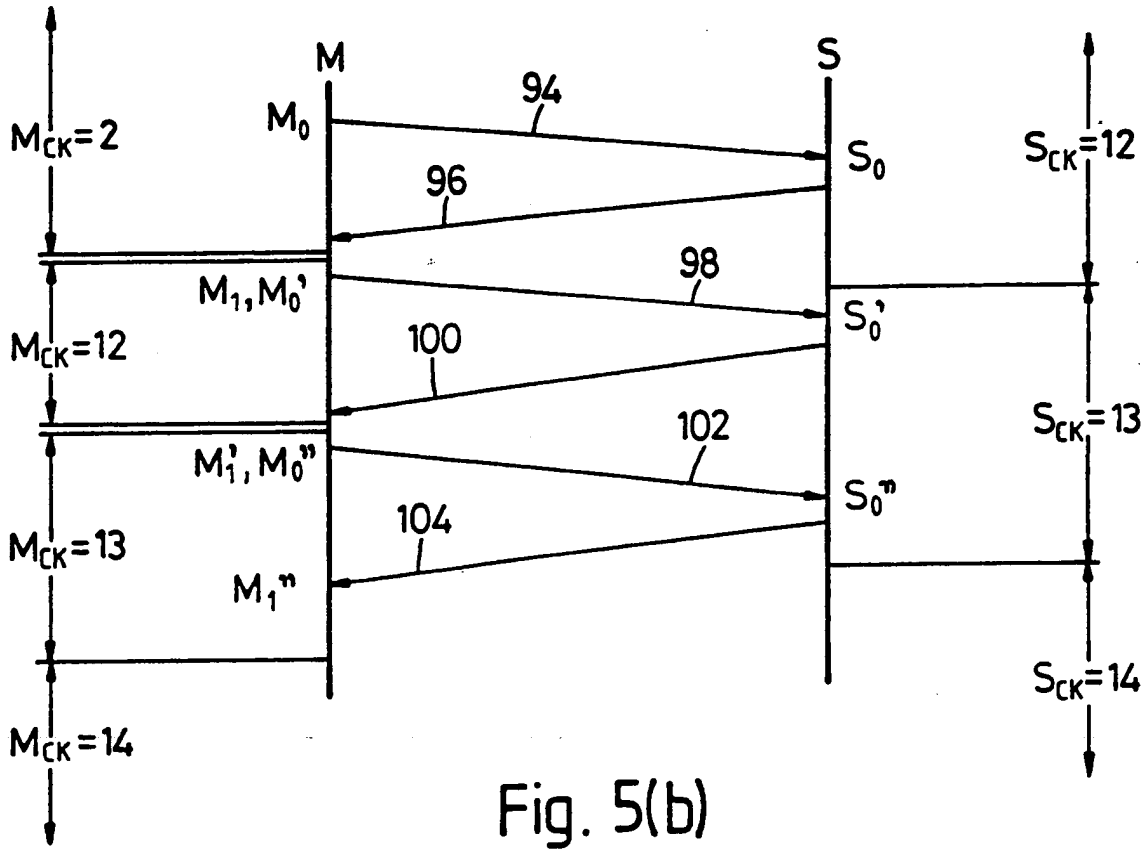
M
S
94
$M_{CK}=2$
M_0
S_0
$S_{CK}=12$
96
98
M_1, M_0'
S_0'
$M_{CK}=12$
100
$S_{CK}=13$
102
M_1', M_0''
S_0''
104
$M_{CK}=13$
M_1''
$S_{CK}=14$
$M_{CK}=14$

Fig. 5(b)

TIMER SYNCHRONISATION SYSTEM

The present invention relates to timer synchronization systems, and in particular to the synchronisation of a current time value between two processors in a computer system.

It is a common requirement for two or more processors in communication with each other to be provided with a clock with which to provide time-stamping information to certain processes, and in certain applications, it is necessary that these clocks are synchronised throughout the system. An example of such time-stamping requirements is found in data communication networks where data packets transferred over the networks may have expiry times placed upon them, to ensure that old data packets containing stale data may be eliminated.

It is also commonly a requirement in synchronising individual clocks to one another that no clock in the system be permitted to decrease its present time which avoids any possibility that time-stamps generated at a later real time may precede earlier ones.

It is therefore an object of the present invention to provide a timer synchronisation scheme which allows individual processors to synchronise clocks while ensuring that any clock may only be adjusted in one direction, ie. only forward in time, or backward in time.

According to one embodiment of the present invention there is provided a method for the synchronisation of a first and a second timer including the following steps:

a) said first timer (M) issuing to said second timer (S) the current value of said first timer (M_0);

b) said second timer (S) comparing said first timer value (M_0) with the current value of said second timer (S_0);

c) if said second timer value (S_0) is less than said first timer value (M_0), said second timer (S) incrementing said current second timer value (S_0) to a new second timer value (S_1) equal said first timer value (M_0); and

d) if said second timer value (S_0) is greater than said first timer value (M_0), said second timer (S) issuing to said first timer (M) the current value of said second timer (S_0).

Embodiments of the present invention will now be described by way of example and with reference to the accompanying drawings in which:

Figure 1 shows a schematic diagram of a routing system suitable for using the present invention, and useful in the description thereof;

Figure 2 shows a flow diagram representing an embodiment of the present invention;

Figure 3 shows a timing diagram representing a one-transaction timing synchronisation operation according to the invention;

Figures 4(a) and 4(b) show two timing diagrams representing respectively a one- and a two-transaction timing synchronisation operation according to the present invention; and

Figures 5(a) and 5(b) show two timing diagrams representing respectively a two- and a three-transaction timing synchronisation operation according to the present invention.

With reference to figure 1 there is shown a routing system 10 for the routing or bridging of data packets on a data communication network. Such a system is described for the purposes of illustrating a typical use of the timer synchronisation scheme.

The routing system 10 includes a number of linecards (only two illustrated) 12,14 each of which is connected to a corresponding network 13,15 of various types. The linecards 12,14 are each responsible for interfacing with the particular type of network to which they are attached. For example, linecard 12 may interface with an FDDI optical fibre network 13, and linecard 14 might interface with an Ethernet type network 15. The linecards 12,14 are mutually connected over a high speed bus 24.

Each line card 12,14 includes an interface unit 30 and a packet processing unit 32. The interface unit 30 is operative to receive data packets from its respective network 13,15 and to place data packets onto its respective network 13,15. Data packets received by interface unit 30 from the network

13 are passed to the packet processing unit 32 for routing of the packets to various addresses throughout the system over the bus 24. Data packets received from other parts of the routing system 10 are received by packet processing unit 32 from the bus 24, and passed to interface 30 for onward transfer onto the network 13. Data packets are typically buffered at various points throughout the system, and may be held in memory pending the outcome of other processing operations, and thus data packets may exist within the system for some time.

In each of the linecards, the interface unit 30 and packet processing unit 32 each operate within their own separate time domain, and there is no communication therebetween, other than by the transfer of data packets. Each unit 30,32 includes an absolute time clock or timer 31,33 so that timing information may be included within packets of data therebetween. In the particular protocol used, data packets received over the network must not be allowed to exist in the system after a certain time period to ensure that stale data is not transmitted around the system. To facilitate this, data packets received by the interface unit 30 are "time-stamped" with an expiry time, which is generated from the absolute time clock therein. These packets are passed to the packet processor where they will be buffered until the packet processor is capable of handling them. Likewise, packets processed for sending over the networks 13,15 by processor 32 are stamped with an expiry time by the packet processor to ensure that the interface unit 30 does not transmit old data packets.

Time-stamp information is checked at entry and exit of the processor unit 32 and interface unit 30. Any packet received after its expiry time is deemed to contain stale data and is therefore to be discarded.

For the time-stamping mechanism to operate correctly, the absolute time clocks of the receiver and the transmitter of the data packet must be synchronised to the same time value. Time value quantization units are chosen to be substantially larger than a transmission time of a data packet between the timing domains, but sufficiently small to provide the required resolution in the ageing out process. In the present illustrative embodiment, the time values of each individual clock are incremented in time quantization units of 10 milliseconds, a typical data packet transmission time is 100 microseconds and expiry times of data packets are typically of the order of 500 milliseconds.

The protocol required for keeping the time value synchronised between the two processors must also cope with the restriction that time values in any particular processor may only be increased, and not decreased. This ensures that packets exceeding the age of 500 milliseconds will always be discarded, at the expense of ageing out of some packets slightly prematurely. Of course, this protocol could be reversed, and it will become evident from the following description, that the present invention may be applied to both protocols.

The protocol must also take into account that the synchronisation operation must take place using the data packet communication channel, the characteristics of which are that delivery time of any packets containing

synchronisation transactions to the destination processors cannot be guaranteed. That is to say, where a data packet is passed from a first processor to a second processor which are, in fact synchronized, the packet may arrive at the second processor at a time later than the departure time of the packet from the first processor. Such a delay may arise from two separate causes: firstly the "time of flight" of the packet (transmission time on the communication channel), and secondly, the buffering delay before processing the packet. In the present illustrative embodiment, this non-deterministic nature of the packet transfer time is substantially due to the uncertainty introduced by the buffering of data packets in both the packet processing unit 32 and the interface unit 30 when transferring data packets therebetween. In practice, the interface unit 30 receives a packet from the network 13, time stamps the packet, and passes it on to the packet processing unit 32, where it may reside in a buffer until it can be processed. The reverse situation also holds: a packet being routed onto the network 13 is processed and time stamped by packet processing unit 32 and passed to interface 30 where it will reside in a buffer until it can be processed by the interface unit for transmission onto the network 13. Because the synchronising transactions are transferred in similar fashion, the synchronisation scheme must take into account this non-deterministic transfer time. Of course, it will be recognized that the same criteria apply to transactions between processors where the "time of flight" of the packet between the processors may form a substantial part of the delay.

In the synchronisation scheme according to the present invention, the processor, or timer which initiates the synchronisation operation is hereinafter referred to as the master M and the processor responding to the synchronisation operation is referred to as the slave S. The identification of which processor is to be master or slave is arbitrary to the present invention, and may be determined by many other factors.

With reference to figure 2, the synchronisation operation is initiated by the master according to some predetermined criteria. For example, the synchronisation may take place at routine intervals. At the outset of the synchronisation operation, the master M issues to the slave S, a data packet containing a synchronisation request and its current time value M_0 (step 50). The master M then awaits a response from slave S.

The slave S receives the synchronisation request and compares the issued master time value M_0 with its own current slave time value S_0 (step 52). Three possible outcomes may arise from this comparison.

Firstly, if the times are the same ($M_0=S_0$), then no synchronisation is necessary, and acknowledgement of this by the slave S to the master M is effected by sending a data packet containing a confirm response and the values M_0, S_0 (step 54). Of course, in the intervening period between sending out the synchronisation request, and receiving the acknowledgement from the slave S, it is possible that the master M has incremented its clock. Thus master M checks its new present value M_1 against the issued and returned value M_0 (step

56). If $M_0=M_1$ then the master clock has not incremented in the intervening period and the synchronisation operation has successfully completed (step 58) with one transaction. If $M_0<>M_1$ then the clock has incremented in the intervening period and the synchronisation transaction is completed, but the synchronisation operation has failed. A further transaction must be re-initiated (return to step 50).

Secondly, in the event that the slave S is behind the master M in time ($M_0>S_0$), then the slave S increments its timer value to a new value S_1 equal to the issued master time value M_0. It acknowledges this action by sending a data packet to the master M containing a confirm response including the received master time value M_0, and the old slave timer value S_0 before synchronisation (step 60). Master M then recognises that slave S has synchronised to the issued master time value M_0 (and can identify by how much if necessary), and checks to see that this time value M_0 is still equal to the present master time value M_1 (step 56). If $M_0=M_1$, then the synchronisation transaction has successfully completed the synchronisation operation (step 58). If during the transaction, master M has incremented its timer such that $M_1>M_0$, then the synchronisation transaction has completed but has failed to complete the synchronisation operation, and a new transaction must be initiated (return to step 50).

Thirdly, in the event that the slave S is ahead of the master M in time ($M_0<S_0$), then the slave is unable to turn back the slave timer due to the

protocol constraint already identified. Slave S therefore acknowledges the synchronisation request by issuing to the master M a data packet containing a confirm response and issued master time M_0 together with the current time value S_0 of the slave timer S (step 62). The master M receives the time value S_0 and increments its timer value M_1 to equal S_0 (step 64). Because the slave S may, in the intervening period, have incremented its time value to a new value S_1, the master automatically initiates a new synchronisation transaction (return to step 50).

The synchronisation operation is alternatively represented in figures 3,4 and 5 as timing diagrams. On the left hand side is shown the time M_{CK}, in arbitrary units, of the master M. The clock interval, or "tick", is represented by horizontal lines at each point where the clock self-increments. Where the master clock has a "set" time imposed upon it by the slave response, this is represented by a double horizontal line. On the right hand side is shown the time S_{CK}, in the same arbitrary units, of the slave S. The clock interval is represented by horizontal lines at each point where the clock self-increments. Where the slave clock has a "set" time imposed upon it by the master M, this is represented by a double horizontal line. Each transaction of two data packets is represented by a continuous line between master and slave (the "request" packet), and a second continuous line returning from slave to master (the "response" packet). The angle of the lines is merely diagrammatic to represent the real packet transfer time (eg. δ_1; fig. 3) between the sending of the

associated data packet, and the processing of that packet by the receiving processor. This time delay δ_1 is, as explained above, a variable, and is composed of a small "flight time" element, and a very much larger buffer time element.

Figure 3 represents a first case where time $M_0=S_0$ and only one transaction is required to successfully complete the synchronisation operation. The master M issues a synchronisation request data packet 70 at a real time t. The packet 70 arrives for processing at slave S at a real time $t+\delta_1$, δ_1 representing the packet transfer time. The slave processes the packet 70, and issues a response packet 72 at time $t+\delta_2$, the difference $\delta_2-\delta_1$ representing the processing time. The packet 72 arrives for processing by the master M at time $t+\delta_3$, the difference $\delta_3-\delta_2$ representing the packet transfer time. The packet 70 is issued by the master indicating its clock time $M_0=20$. The slave receives the packet 70, compares the master time $M_0=20$ with its own time $S_0=20$ and thus confirms time set in response packet 72. Master M receives this response at time $M_1=20$ which confirms that the synchronisation operation has completed with a single transaction.

Figure 4 represents a second case where $M_0>S_0$, and shows in figure 4(a) the situation where a single transaction is required to complete the synchronisation operation, and in figure 4(b) the situation where two transactions are required to successfully complete synchronisation, ie. the master increments before receiving the slave response.

In figure 4(a), master M issues request data packet 74 at time $M_0=20$. This is received by slave S at $S_0=15$. Slave S therefore sets clock S_{CK} so that $S_1=20$ and issues response packet 76 to master M with this set time. Master M receives this packet at time $M_1=20$ which confirms that the synchronisation operation has completed with a single transaction.

In figure 4(b), the operation commences in like manner with issue of request packet 78 by master M at $M_0=20$. However, by the time the response packet 80 from S has been received, the clock M_{CK} has self-incremented, and $M_1=21$. A second transaction is then necessary and is implemented with issuance of packet 82 at $M_0'=21$, receipt at S_0' and return of response packet 84 in the manner of figure 4(a). The synchronisation operation is then completed.

Figure 5 represents the third possible case where $M_0<S_0$, showing in figure 5(a) a situation where two transactions are required to complete synchronisation, and in figure 5(b) a situation where three transactions are required to successfully complete synchronisation, ie. the slave self-increments during synchronisation of the master to the slave time value. The second transaction 98,100 is indicated with M_0', S_0', M_1' and S_1' and the third transaction 102,104 is indicated with M_0'', S_0'', M_1'' and S_1''.

In figure 5(a), master M issues synchronisation request packet 86 at $M_0=2$. Slave S receives packet 86 at time $S_0=12$, and issues response packet 88 indicating time $S_0=12$. Master M receives this packet, and increments its

time $M_1=12$. It then starts a new transaction with packet 90, indicating a new time $M_0'=12$. Slave S receives this at time $S_0'=12$, and confirms this in response packet 92. Master M receives the response at time $M_1'=12$ which confirms that the synchronisation transaction has been completed.

In figure 5(b), the procedure is the same as that described for figure 5(a) with respect to a first transaction 94,96, but before the second transaction synchronisation request packet 98 is received by slave S, the slave clock has self-incremented so that the packet 98 is received at time $S_0'=13$. This is indicated in response packet 100, and thus the master M must initiate a further transaction 102,104 in order to successfully complete the synchronisation operation.

The timing diagrams of figures 3, 4 and 5 are exemplary only, and do not embrace all variations of the synchronisation scheme herein described.

Synchronisation requests are initiated at predetermined routine intervals, the interval allowing for the degree of synchronal accuracy of the two clocks to be synchronised. A further check may be made to establish whether the synchronisation transactions are causing clock changes too frequently, suggesting a system fault.

It will be recognised that the present invention may be adapted to operate for the protocol where master and slave clocks may only be decreased in value. This corresponds to the situation where data packets must be allowed to exist for a predetermined period of time, at the expense of ageing out some packets

slightly tardily.

It will be recognised that the present invention applies to any system in which two processors require synchronisation, and is not restricted to the embodiments herein described in respect of a routing system.

CLAIMS

1. A method for the synchronisation of a first and a second timer including the following steps:

a) said first timer (M) issuing to said second timer (S) the current value of said first timer (M_0);

b) said second timer (S) comparing said first timer value (M_0) with the current value of said second timer (S_0);

c) if said second timer value (S_0) is less than said first timer value (M_0), said second timer (S) incrementing said current second timer value (S_0) to a new second timer value (S_1) equal said first timer value (M_0); and

d) if said second timer value (S_0) is greater than said first timer value (M_0), said second timer (S) issuing to said first timer (M) the current value of said second timer (S_0).

2. A method for the synchronisation of a first and a second timer including the following steps:

a) said first timer (M) issuing to said second timer (S) the current value of said first timer (M_0);

b) said second timer (S) comparing said first timer value (M_0) with the current value of said second timer (S_0);

c) if said second timer value (S_0) is greater than said first timer value

(M_0), said second timer (S) decrementing said current second timer value (S_0) to a new second timer value (S_1) equal to said first timer value (M_0); and

d) if said second timer value (S_0) is less than said first timer value (M_0), said second timer (S) issuing to said first timer (M) the current value of said second timer (S_0).

3. A method according to claim 1 or claim 2 wherein if said first timer value (M_0) is equal to said second timer value (S_0), said first timer (M) completing the synchronisation process.

4. A method according to claim 1 or claim 2 wherein if said first timer value (M_0) is equal to said second timer value (S_0), said first timer (M) comparing said first timer value (M_0) with its latest current value (M_1), and if said first timer value (M_0) is not equal to said latest current value (M_1), returning to step a).

5. A method according to any previous claim further including the step:

e) if the condition in c) holds, said second timer (S) issuing to said first timer (M) the value of said second timer value (S_0) and said first timer value (M_0).

6. A method according to claim 5 further including the step:

f) said first timer (M) comparing said first timer value (M_0) with its latest current value (M_1), and if said first timer value (M_0) is not equal to said latest current value (M_1), returning to step a).

7. A method according to any one of claims 1 to 4 further including the step:

e) if the condition in d) holds, said first timer (M) altering said first timer current value (M_0) to a new first timer value (M_1) equal to said second timer value (S_0) and returning to step a).

8. Apparatus for the synchronisation of two processors including:

a first processor including a first timer;

a second processor including a second timer;

means for said first processor to issue to said second processor the current value of said first timer (M_0);

means for said second processor to compare said issued first timer value (M_0) with the current value of said second timer (S_0);

means for said second processor to increment said current second timer value (S_0) to a new second timer value (S_1) equal said first timer value (M_0) if said second timer value (S_0) is less than said first timer value (M_0); and

means for said second processor to issue to said first processor the current value of said second timer (S_0) if said second timer value (S_0) is greater

than said first timer value (M_0).

9. Apparatus for the synchronisation of two processors including:

a first processor including a first timer;

a second processor including a second timer;

means for said first processor to issue to said second processor the current value of said first timer (M_0);

means for said second processor to compare said issued first timer value (M_0) with the current value of said second timer (S_0);

means for said second processor to decrement said current second timer value (S_0) to a new second timer value (S_1) equal said first timer value (M_0) if said second timer value (S_0) is greater than said first timer value (M_0); and

means for said second processor to issue to said first processor the current value of said second timer (S_0) if said second timer value (S_0) is less than said first timer value (M_0).

10. A method for the synchronisation of a first and a second timer substantially as described herein and with reference to the accompanying drawings.

Patents Act 1977
Examiner's report to the Comptroller under Section 17 (The Search Report)

Application number
GB 9220729.9

Relevant Technical fields

(i) UK Cl (Edition L) H4P (PPF, PSEX, PSX, PDRX); H4K (KTA, KTX); G4A (AFT)

(ii) Int Cl (Edition 5) H04L 7/00, 7/10, 12/28, 12/42; G06F 13/42

Search Examiner
K WILLIAMS

Databases (see over)
(i) UK Patent Office

(ii)

Date of Search
24 AUGUST 1993

Documents considered relevant following a search in respect of claims 1-9

Category (see over)	Identity of document and relevant passages	Relevant to claim(s)
X	EP 0135764 A2 (I B M) See page 12, lines 1-2	1,8

Identity of document and relevant passages	Relevant to claim(s)

Categories of documents

X: Document indicating lack of novelty or of inventive step.

Y: Document indicating lack of inventive step if combined with one or more other documents of the same category.

A: Document indicating technological background and/or state of the art.

P: Document published on or after the declared priority date but before the filing date of the present application.

E: Patent document published on or after, but with priority date earlier than, the filing date of the present application.

&: Member of the same patent family, corresponding document.

Databases: The UK Patent Office database comprises classified collections of GB, EP, WO and US patent specifications as outlined periodically in the Official Journal (Patents). The on-line databases considered for search are also listed periodically in the Official Journal (Patents).